

4.3.2 Special Relational Operations:- (Selection, projection, join)

Selection:

The algebraic selection operator(not to be confused with SQL SELECT) yields a "horizontal" subset of a given relation-that is, that subset of tuples within the given relation for which a specified predicate is satisfied. The predicate is expressed as a Boolean combination of terms, each term being a simple comparison that and be establishes as true or false for a given tuple by inspecting that tuple in isolation.

Example:-

1- S WHERE City = 'London'



S#	Sname	status	City
S1	Smith	20	London
S4	Clark	20	London

2- P WHERE Weight < 14



P#	Pname	Color	Weight	City
P1	Nut	Red	12	London
P5	Cam	blue	12	Paris

3- SP WHERE S# = 'S1' AND P# = 'P1'

S#	P#	QTY
S1	P1	300

Fig. (4.4) Three sample selections.

Projection

The projection operator yields a "vertical" subset of a given relation by selecting specified attributes in a specified left- to- right order and the eliminating duplicate tuples within the attributes selected. Projection provides us with a way to reorder the attributes of a given relation.

No attribute may be specified more than once in a projection operator.
 $R[X,Y,\dots,Z]$.

Examples:-

1- $S[\text{city}]$

City
London
Paris
Athens

2- $S[\text{Sname, city, S\#, status}]$

Sname	city	S#	Status
Smith	London	S1	20
Jones	Paris	S2	10
Blake	Paris	S3	30
Clark	London	S4	20
Adams	Athens	S5	30

3- $(S \text{ Times } P) [\text{status, color}]$

Status	Color
20	Red
10	Red
30	Red
20	Green
10	Green
30	Green
20	Blue
10	Blue
30	Blue

Fig. (4.5) Three sample projections.

Join

Two tables each have a column defined over some common domain, they may be joined over those two columns, the result of the join is a new, table in which each row is formed by concatenating two rows, one from each of the original tables, such that the two rows have the same value in those two columns.

Example: tables S and P may be joined over their city column and the result is shown as follows:-

S#	Sname	Status	S.city	P#	Pname	Color	weight	P.city
S1	Smith	20	London	P1	Nut	Red	12	London
S1	Smith	20	London	P4	Screw	Red	14	London
S2	Jones	10	Paris	P2	Bolt	Green	17	Paris
S3	Blake	30	Paris	P2	Bolt	green	17	Paris

Equijoin:- is that join in which the " joining condition" is based on equality between values in the common column, and the result of the equijoin must include two identical attributes. (as shown in the above example).

Natural join:- is an equijoin with one of the identical columns eliminated.

Note:- join is not a primitive operation it is always equivalent to taking the extended Cartesian product of the two relations and the performing an appropriate restriction (selection) on the result.

Example 1 :- the natural join is a projection of a restriction of a product. The following expression represents the natural join of relation S on S# with relation SP on S#

((S Times SP) where S.S# = SP.S#)

[S.S# , Sname,Status, city,P#, Qty]

Which is the same as the following expression:-

S JOIN SP always means natural Join

Result for both of the above expression is:-

S.S#	S.Sname	S.status	S.city	SP.P#	SP.QTY
S1	Smith	20	London	P1	300
S1	Smith	20	London	P2	200
S1	Smith	20	London	P3	400
S1	Smith	20	London	P4	200
S1	Smith	20	London	P5	100
S1	Smith	20	London	P6	100
S2	Jones	10	Paris	P1	300
S2	Jones	10	Paris	P2	400
S3	Blake	30	Paris	P2	200
S4	Clark	20	London	P2	200
S4	Clark	20	London	P4	300
S4	Clark	20	London	P5	400

Fig. (4.6) JOIN of S and SP over S# (S JOIN SP).

Example 2:- The greater than JOIN can be expressed by using:-

(S TIMES P) where S.city > P.city

Result of the above expression is as follows:-

S#	Sname	Status	S.city	P#	Pname	Color	weight	P.city
S2	Jones	10	Paris	P1	Nut	Red	12	London
S2	Jones	10	Paris	P4	Screw	Red	14	London
S2	Jones	10	Paris	P6	Cog	Red	19	London
S3	Blake	30	Paris	P1	Nut	Red	12	London
S3	Blake	30	Paris	P4	Screw	Red	14	London
S3	Blake	30	Paris	P6	Cog	Red	19	London

Database

Notes:-

- 1- A JOIN B is identical to A TIMES B when A and B have no common attributes.
- 2- We allow a sequence of JOINS to be written without embedded parentheses.

Ex :- The two expression :- (A JOIN B) JOIN C

A JOIN (B JOIN C)

Can be written as

A JOIN B JOIN C

Since, the JOIN is associative.

Division

(DIVIDEBY) operator takes two relations, one binary and one unary and builds a relation consisting of all values of one attribute of the binary relation, such that the other attribute in the binary relation match all values in the unary relation.

A DIVIDEBY B

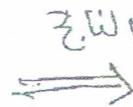
Example:-

Dend

S#	P#
S1	P1
S1	P2
S1	P3
S2	P1
S2	P2
S3	P1
S3	P4

DoR

P#
P1
P2



Dend DivideBy DoR

S#
S1
S2

Fig. (4.7) sample divisions.

Database

Relational languages:

- 1- **Relational algebra:** is theoretical language with operations that work on one or more relations to define another relation without changing the original relations. Such as projection, products, union, join, division.
- 2- **Relational calculus:** it specifies what is to be retrieved rather than how to retrieve it. (Range of value)
- 3- **Database language (SQL):** should allow a user to
 - Create the database and relation structure
 - Perform basic data management tasks, such as the insertion, modification, and deletion of data from the relations.
 - Perform both simple and complex queries

SQL: can be used by a range of users including database administrators (DBA), management personal, application programmers, and many other types of end-user.

Relational language

- SQL (standard query language)
- QBE (query by example)

SQL statement

- **DDL (data definition language)**
 - 1- Create table
 - 2- Alter table
 - 3- Drop table

Ex: create database travel and build table name car, owner, and data of travel with all columns.

Car	Owner	Data travel
car-no	name	travel-no [pk]
model	address	date
color	car-no	car-no [fk] Day of travel

- Create database travel
- Create table cars (car-no char (8), model char (10), color char (8))
- Create table owner (name char (10), address char (20), car-no char (8))
- Create table data travel (travel-no numeric (10), date date, car-no char (8), days of travel numeric (3))

Database

- Insert into car values ("1134567", "Toyota", "white")
- Insert into owner values ("ahmed", "Baghdad", "1134567")
- Insert into data travel (234, {10/10/2003}, "1134567")

Add column to table car named motor no

- Alter table car add column motor no char (20)

Delete column from table car named color

- Alter table car drop column color

Assign the [pk] and [fk]

- Alter table car add constraint pp primary key (car-no);
- Alter table data of traveling add constraint ff foreign key (car-no)

- **DML (data manipulated language DB data)**

- 1- Select
- 2- Insert
- 3- Delete
- 4- Update

- Select * from car

Get all car information

- Select car. * from car

Get all car-no

- Select car. car-no from car

Get car model for car-no (657489)

- Select car. model, car-no from car where car. car-no="657489"

Get cars with Toyota models

- Select car. Model, car. car-no from car where car. model="Toyota"

Delete each car when car-no 657489

- Delete from car where car. car-no="657489"

Get all cars model when color is white and owner "ahmed"

- Select car. * from car where car. Color="white"
and owner.name ="ahmed"

How many trip the car-no (4455986) had traveled

- Select count (*) from travel where car-no="4455986"

Database

- Select count (*) from travel where data travel. date="1/1/2007"

How many days car-no (1005) had traveled

- Select sum (days of travel) from travel
Where data travel. car-no="1005"

Note: keyword (distinct) means getting the record without duplicate.

- Select distinct data travel. car-no from travel

Note: retrieval in order ascending and descending

Get all trips by car (1005) from the latest one

- Select data travel. car-no, data travel. date, data travel.travel-no
From travel where data travel.car-no="1005"
Order by date ASCE

How many each car had traveled

- Select sum (day of travel) from data travel group by data travel. car-no

Change the driver of car 1005 to ali how lived in karada

- Update owner set name="ali", address="karada" car-no="1005"
Where owner. car-no="1005"

Delete the driver ahmed from company

- Delete from owner where owner.name="ahmed"

Remove all record from owner

- Delete from owner